Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 154 (2006) 802–817

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

First vs. best improvement: An empirical study

Pierre Hansen*, Nenad Mladenović[1]

GERAD and HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Qué., Canada H3T 2A7

Received 4 February 2003; received in revised form 18 February 2004; accepted 31 May 2005
Available online 13 December 2005

Abstract

When applying the 2-opt heuristic to the travelling salesman problem, selecting the best improvement at each iteration gives worse results on average than selecting the first improvement, if the initial solution is chosen at random. However, starting with 'greedy' or 'nearest neighbor' constructive heuristics, the best improvement is better and faster on average. Reasons for this behavior are investigated. It appears to be better to use exchanges introducing into the solution a very small edge and fairly large one, which can easily be removed later, than two small ones which are much harder to remove.
© 2005 Published by Elsevier B.V.

Keywords: Travelling salesman; Heuristic; Metaheuristic; Variable neighborhood search

1. Introduction

Greedy heuristics for combinatorial optimization select at each iteration the feasible move which gives the best improvement in objective function value. It is well known that for several problems such heuristics are optimal. This is, for instance, the case for the Minimum Spanning Tree problem [8]. However, for many other combinatorial optimization problems the greedy heuristic is not optimal anymore. But it is still intuitively appealing: if a best improvement is selected at each iteration one would expect that a better local optimum would be found than with some other selection rule (which might be preferred because finding the best improvement may be time consuming). The main purpose of the experimental study reported here is to refute, by example, such conventional wisdom. We consider heuristic solution of the Travelling salesman problem (TSP) [7], i.e., given n cities and distances between pairs of them find a shortest tour passing once and only once through each city. If the initial solution is chosen at random, the first improvement is better and faster than the best improvement. However, if the initial solution is found by some constructive heuristic, i.e., if the initial solution is not too bad, then the best improvement is slightly better and even faster in average than the first improvement. Indeed, it is shown below that when applying the 2-opt heuristic [2,3], to both the Euclidean and random distance TSP, using first improvement instead of best improvement gives *on average* local optima with a smaller value, but only if initial solution is chosen at random. This result was obtained by serendipity when applying the Variable Neighborhood Search (VNS) metaheuristic [9,5,6] to the TSP. VNS exploits systematically changes of neighborhood both in the descent phase (Variable Neighborhood Descent, VND) which leads to a local optimum and in

* Corresponding author. Fax: +1 514 3405665.
E-mail addresses: pierre.hansen@gerad.ca (P. Hansen), nenad.mladenovic@gerad.ca (N. Mladenović).
[1] Present address: School of Mathematics, Brune University, UK.

0166-218X/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.dam.2005.05.020

the exploratory phase (or VNS proper) which seeks a better local optimum. For the TSP, VND uses a series of heuristics, in order of increasing complexity: 1-opt, 2-opt, 2.5-opt, Or-opt, etc.

Variants of the 2-opt heuristic used in this study are described in the next section. First, experiments and computational results are presented in Section 3. Several further series of experiments designed to find explanations of the observed phenomenon are described, with their results, in Section 4. Conclusions are drawn in the last section.

We are, of course, aware that the 2-opt heuristic is not sufficient, taken alone, even with a streamlined implementation, to provide good quality near-optimal solutions to the TSP. The focus of this paper is *not* on building a better heuristic (the first improvement neighbor-list implementation is indeed not new, see e.g. [7] although perhaps not all variants considered below have yet been studied) but on *studying a surprising phenomenon* as one ingredient. The insight so obtained will, hopefully, prove to be quite general and therefore useful in the design of new heuristics (or more efficient versions of existing ones) for the TSP and other combinatorial optimization problems.

2. Heuristics

The 2-opt heuristic for the TSP [2,3] removes at each iteration a pair of edges in the tour and reconnects their endpoints in the only other way which gives a connected tour. This is done as long as the length of the tour decreases. The simplest, and classical, implementation of the 2-opt heuristic consists in a systematic enumeration of pairs of edges, to be considered for deletion, along the tour. In the *best improvement* version, all $n(n-1)/2$ such pairs must be checked at each iteration which thus requires $O(n^2)$ time for a *n*-city problem. Detailed rules of this heuristic, noted BI-CL, are presented in Fig. 1, where the simplest so-called *array* data structure for tour representation is used (for different data structures which appear to be more efficient in solving very large problem instances, see for example [4]). It is time consuming, and dominated by the neighbor-list implementation, described below. In the *first improvement*

```
1. Data. Input number $n$ of cities, matrix $D = (d_{ij})$ of distances between pairs of
   cities.

2. Initial solution. Draw a tour $T$ at random, let $x_1, x_2, \ldots, x_n$ denote the indices
   of successive cities (or nodes) visited, let $x_{n+1} = x_1$. Compute the value $v(T)$ of
   $T$:
        $v(T) = \sum_{i=1}^{n} d_{x_i, x_{i+1}}$.

3. Best improvement. Let $\Delta = 0$
   For $i = 1, \ldots, n-2$
       If $i = 1$ then $n' = n-1$, else $n' = n$;
       For $j = i+2, \ldots, n'$
           Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i+1},x_{j+1}} - d_{x_i,x_{i+1}} - d_{x_j,x_{j+1}}$
           If $\Delta_{ij} < \Delta$, let $\Delta := \Delta_{ij}$, $i^* = i, j^* = j$;
       Endfor
   Endfor
   If $\Delta = 0$, go to 5;
   Let $i = i^*$ and $j = j^*$.

4. Update. Modify tour by deletion of edges between the $i^{th}$ and $i+1^{th}$ nodes and
   the $j^{th}$ and $j+1^{th}$ nodes, and insertion of edges between the $i^{th}$ and $j^{th}$ nodes
   and between the $i+1^{th}$ and $j+1^{th}$ nodes:
   Exchange position of elements $i+\ell$ and $j-\ell+1$ in array $x$, for $\ell = 1, \ldots, \lfloor j-i \rfloor/2$;
   $v(T) := v(T) + \Delta$;

5. Best tour found. Output $v(T)$ the value of the best tour found and the corre-
   sponding list $x_1, \ldots, x_n$ of indices of successive cities visited.
```

Fig. 1. Heuristic BI-CL: classical implementation of heuristic 2-opt with best improvement criterion.

```
3'. First improvement. Let $\Delta = 0$
 For $i = 1, \dots, n-2$
        If $i = 1$ then $n' = n-1$ else $n' = n$;
        For $j = i+2, \dots, n'$
              Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i+1},x_{j+1}} - d_{x_i,x_{i+1}} - d_{x_j,x_{j+1}}$
              If $\Delta_{ij} < 0$ go to 4;
        Endfor
 Endfor
 Go to 5.
```

Fig. 2. Heuristic FI-CL: modified step $3'$ in heuristic BI-CL for classical implementation of 2-opt with first improvement criterion.

version enumeration of pairs of edges is interrupted as soon as a 2-opt exchange decreasing the length of the tour is found (see Fig. 2). The time per iteration remains in $O(n^2)$ in worst case and, indeed, at the last one all $n(n-1)/2$ pairs will be checked to show a local optimum has been reached. Average time per iteration will be substantially less in practice.

Modification to the BI-CL heuristic to obtain a classical first improvement version noted FI-CL are presented in Fig. 2.

The property (first noted by Steiglitz and Weiner [11]) that, for a 2-opt exchange to reduce the current tour length, one of the entering edges must be shorter than one of the leaving edges leads to further and better implementations. A preliminary step is then to rank all edges incident with each node by order of non-decreasing length. Enumeration is done by examining potential entering edges incident at a node, say the ith one in that order, and with length smaller than that of the edge from that ith node to the next one in the tour. Initial nodes i may be enumerated in any order, e.g., the natural one, or that of their position along the tour. As all possibilities must be checked, it is necessary to consider also entering edges from the ith to the jth node in the tour with length smaller than that of the edge in the tour from their second node, i.e., the jth one, to the next. It is equivalent to do the search along successive nodes along the tour and then reverse the process. Details of an implementation doing this are given in Fig. 3 for the best improvement criterion. The corresponding heuristic is noted BI-NL. Modifications to obtain a version following the first improvement criterion are presented in Fig. 4. A few further variants are considered in Section 4 and were designed in order to better understand the results of a comparison between the versions of the 2-opt heuristic presented here. These results will be summarized next.

3. First experiments and results

All programs are written in Fortran 90 and experiments conducted on a Sun Ultra I computer with 143 MHz UltraSparc processor.

The heuristics BI-NL, FI-CL and FI-NL were first applied to a series of Euclidean travelling salesman problem with $n = 20$–1000 cities. To define these problems, n points were randomly generated from a uniform distribution on the $[0, 100] \times [0, 100]$ square. Distances $D = (d_{ij})$ are Euclidean. Heuristic BI-CL is not included in the comparison since at each of its steps the same exchange as in BI-NL takes place and computing time is larger (multiplied roughly by 1.5 for problems with $n = 500$). Results are presented in Table 1. They are averages over 1000 instances for small problems (up to $n = 150$) and over 100 instances for larger ones ($n = 200$–1000). It appears that:

(i) Except for very small problems ($n = 20$ and 30), FI-CL performs better than BI-NL on average. The difference in solution values is substantial and about 2% for the larger problems. (As a yardstick for comparison recall that Johnson and McGeoch [7] consider as substantial a 0.5% improvement when going from 2-opt to 2.5-opt [1] for random Euclidean instances as well.)

(ii) Heuristic FI-NL is always better than BI-NL and more so than FI-CL. The difference is about 3.3% for the larger problems, and reaches 3.48% for $n = 1000$. Note that for $n = 1000$, random initial solution and 100 random Euclidean instances, a 7.9% average percent excess of FI-NL 2-opt over the Held–Karp lower bound is reported in [7] (the implementation used there differs slightly from that of the present paper in that it uses truncated neighbor

```
1. Data and preprocessing. Input number $n$ of cities, matrix $D = (d_{ij})$ of distances
   between pairs of cities. For $i = 1, \ldots, n$, rank the edges incident with node $i$ by
   order of non-decreasing $d_{ij}$ (ties being broken by an arbitrary rule, e.g., smallest
   second index). Let $D' = (d'_{ij})$ denote the matrix so obtained, $R = (r_{ij})$ the matrix
   of corresponding indices of ranked edges and $P = (p_{i\ell})$ the matrix which reports
   the position of a city with index $\ell$ in the $i^{th}$ column of matrix $D'$: $p_{i,r_{ij}} := j$,
   $\forall i, j = 1, .., n$.

2. Initial solution. Draw a tour $T$ at random, let $x_1, x_2, \ldots, x_n$ denote the indices
   of successive cities (or nodes) visited, let $x_{n+1} = x_1$. Compute the value $v(T)$ of
   $T$:
   $$v(T) = \sum_{i=1}^{n} d_{x_i, x_{i+1}}.$$
   Find position of each city in the tour: $y_{x_i} := i, \forall i = 1, .., n$;

3. Best improvement. Let $\Delta = 0$
   (a) For $i = 1, \ldots, n$
         $n' = p_{x_i, x_{i+1}} - 1$
         For $j' = 2, \ldots, n'$
               Find position in the tour of the city with index $c := r_{x_i j'}$ as $j := y_c$;
               If $i = j$, continue with next $j'$; else
               Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i+1},x_{j+1}} - d_{x_i,x_{i+1}} - d_{x_j,x_{j+1}}$
               If $\Delta_{ij} < \Delta$, let $\Delta := \Delta_{ij}$, $i^* = i, j^* = j$;
         Endfor
   Endfor
   (b) For $i = 2, \ldots, n+1$
         $n' = p_{x_i, x_{i-1}} - 1$
         For $j' = 2, \ldots, n'$
               Find position in the tour of the city with index $c$
               ($c := r_{x_i j'}$) as $j := y_c$; if $j = 1$, set $j = n + 1$;
               If $i = j$, continue with next $j'$; else
               Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i-1},x_{j-1}} - d_{x_i,x_{i-1}} - d_{x_j,x_{j-1}}$
               If $\Delta_{ij} < \Delta$, let $\Delta := \Delta_{ij}$, $i^* = i - 1, j^* = j - 1$;
         Endfor
   Endfor
   If $\Delta = 0$, go to 5;
   Let $i = i^*$ and $j = j^*$.

4. Update. Modify tour by deletion of edges between the $i^{th}$ and $i + 1^{th}$ nodes and
   the $j^{th}$ and $j + 1^{th}$ nodes, and insertion of edges between the $i^{th}$ and $j^{th}$ nodes
   and between the $i + 1^{th}$ and $j + 1^{th}$ nodes:
   Exchange position of elements $i+\ell$ and $j-\ell+1$ in array $x$, for $\ell = 1, \ldots, \lfloor j-i \rfloor/2$;
   Update array $y$ as $y_{x_\ell} := \ell$, for $\ell = i + 1, \ldots, j$;
   $v(T) := v(T) + \Delta$;

5. Best tour found. Output $v(T)$ the value of the best tour found and the corre-
   sponding list $x_1, \ldots, x_n$ of indices of successive cities visited.
```

Fig. 3. Heuristic BI-NL: neighbor-list implementation of heuristic 2-opt with best improvement.

lists). Thus, BI-NL is about 11.4% over this bound. Improvement curves as function of problem size for FI-CL and FI-NL over BI-NL are given in Fig. 5, where the % improvements are defined as $(v(\text{BI})\text{-}v(\text{FI}))/v(\text{FI}) \times 100$.

(iii) Computing times of all three heuristics increase with problem size, but at different rates: for BI-NL and FI-CL it is approximately multiplied by 10 when n doubles; for FI-NL it is approximately multiplied by about 6 when n doubles. Moreover, the computing time of this last heuristic is much lower than that of the two others, i.e., almost 100 times less when $n = 1000$.

```
3. First improvement.
 (a) For $i = 1, \dots, n$
         $n' = p_{x_i,x_{i+1}} - 1$
         For $j' = 2, \dots, n'$
                 Find position in the tour of the city with index $c := r_{x_i j'}$ as $j := y_c$;
                 If $i = j$, continue with next $j'$; else
                 Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i+1},x_{j+1}} - d_{x_i,x_{i+1}} - d_{x_j,x_{j+1}}$
                 If $\Delta_{ij} < 0$, go to 4;
         Endfor
 Endfor
 (b) For $i = 2, \dots, n+1$
         $n' = p_{x_i,x_{i-1}} - 1$
         For $j' = 2, \dots, n'$
                 Find position in the tour of the city with index $c$
                 ($c := r_{x_i j'}$) as $j := y_c$; if $j = 1$, set $j = n + 1$;
                 If $i = j$, continue with next $j'$; else
                 Compute $\Delta_{ij} = d_{x_i,x_j} + d_{x_{i-1},x_{j-1}} - d_{x_i,x_{i-1}} - d_{x_j,x_{j-1}}$
                 If $\Delta_{ij} < 0$, let $i := i - 1$, $j := j - 1$ and go to 4;
         Endfor
 Endfor
 Go to 5;
```

Fig. 4. Heuristic FI-NL: modified step 3′ to replace step 3 in heuristic BI-NL for neighbor-list implementation of heuristic 2-opt with first improvement criterion.

(iv) The number of iterations of BI-NL is much smaller than those of FI-NL (about 2.3 times less for $n = 1000$) and, most of all, of FI-CL (about 6 times less for $n = 1000$). That the computing time of FI-CL remains smaller despite this discrepancy shows it is much inferior per iteration to that of BI-NL.

To corroborate these results the same three heuristics were applied to a series of Euclidean problems from the TSP-LIB [10], with $n = 51$ to $n = 1432$ cities. While variance is larger than in the previous table, single instances of each size being solved instead of 100 or 1000 (the average error with 10 random initial solutions are reported in Table 2), the conclusions are similar to the previous ones. In addition, optimal values being known for these problems, one can see that residual error of FI-NL is about 7.5% for the larger instances. This is of course substantial, but has to be expected as 2-opt is not a very powerful heuristic for the TSP *when used alone*. The reduction in error for the five largest problems, which have over 1000 cities, i.e., u1060, pcb1173, d1291, rl1323 and u1432, are of 1.94%, 5.22%, 4.33%, 5.55% and 5.13%, respectively. This corroborates the observation that the percentage of improvement augments with problem size.

A third series of experiments consisted in the application of the same three heuristics to randomly generated problems without structure, i.e., such that all distances are uniformly drawn at random in the interval [0, 100]. Results are presented in Table 3. Again results are averages over 1000 instances for $n = 20$–150 and over 100 instances for $n = 200$–1000. Concerning solution values, they differ markedly from the previous ones in two contrasting ways:

(v) For all problem sizes, FI-CL performs worse than BI-NL. The difference in average value of the solution obtained is between 3% and 6% of the best value obtained, in favor of BI-NL.

(vi) For all problem sizes FI-NL performs better than BI-NL and FI-CL. Moreover, differences are much larger than for the Euclidean case: they reach about 20% of the best value found for $n = 100$, almost 50% for $n = 500$ and over 67% for $n = 1000$ (see also Fig. 6).

Conclusions regarding computing times are also different:

(vii) While for Euclidean problems BI-NL and FI-CL used computing times of the same order of magnitude, for randomly generated distances computing times of the latter heuristic are 22 times less than those of the former. Computing times of FI-NL remain by far the smallest and very similar to those used for Euclidean problems.

Table 1
Average results for Euclidean TSP on 1000 random instances, for $n = 20, \ldots, 150$ (average of averages reported in line 'Av. 20–150'), 100 random instances for $n = 200, 250, \ldots, 500$ (line 'Av. 200–500' gives average results) and 100 random instances for $n = 600, 700, \ldots, 1000$ (line 'Av. 600–1000' reports average)

n	Obj. function values			% improv.		CPU times			# of iterations		
	BI-NL	FI-CL	FI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL
20	394.21	395.48	393.94	−0.32	0.07	0.00	0.00	0.00	16	37	24
30	473.86	474.80	472.85	−0.20	0.21	0.00	0.00	0.00	26	69	41
40	541.93	540.52	538.59	0.26	0.62	0.01	0.01	0.00	36	107	58
50	603.50	601.35	598.66	0.36	0.81	0.02	0.02	0.00	46	149	77
60	654.69	652.68	648.95	0.31	0.88	0.04	0.03	0.00	58	192	95
70	704.35	700.05	695.15	0.62	1.32	0.07	0.05	0.01	68	239	115
80	750.91	747.04	740.06	0.52	1.47	0.11	0.07	0.01	79	288	135
90	796.40	790.03	783.40	0.81	1.66	0.16	0.12	0.01	91	338	154
100	836.68	828.75	821.42	0.97	1.86	0.22	0.19	0.01	102	391	176
110	873.51	864.90	857.34	1.00	1.89	0.30	0.27	0.02	114	445	197
120	910.83	903.99	894.46	0.76	1.83	0.40	0.37	0.02	125	502	217
130	948.84	932.20	928.20	1.79	2.22	0.51	0.46	0.02	137	561	241
140	983.70	974.02	964.32	1.00	2.01	0.66	0.62	0.03	149	619	261
150	1016.08	1000.62	994.05	1.56	2.22	0.83	0.81	0.04	161	682	283
200	1166.58	1152.36	1144.30	1.24	1.95	2.08	1.95	0.07	222	988	397
250	1305.99	1279.39	1266.69	2.10	3.10	4.21	4.00	0.12	283	1311	520
300	1421.77	1400.41	1383.51	1.54	2.77	7.66	6.59	0.19	346	1659	630
350	1530.62	1504.22	1489.10	1.77	2.79	12.87	11.30	0.27	410	2015	750
400	1635.59	1603.03	1588.80	2.05	2.94	20.39	16.75	0.38	474	2387	892
450	1727.46	1697.76	1680.46	1.77	2.80	30.30	25.71	0.49	541	2793	1008
500	1821.29	1793.68	1766.90	1.56	3.08	36.80	34.23	0.63	605	3172	1138
600	1994.88	1958.54	1931.68	1.88	3.27	68.53	60.72	1.00	734	3949	1410
700	2152.85	2111.12	2083.07	2.00	3.35	111.52	96.51	1.41	870	4801	1660
800	2292.88	2252.68	2224.27	1.81	3.08	169.45	141.41	1.87	1008	5663	1950
900	2428.38	2383.16	2345.38	1.93	3.54	237.63	203.08	2.42	1144	6526	2236
1000	2564.79	2508.58	2478.63	2.27	3.48	369.11	271.42	3.03	1279	7407	2505
Av. 20–150	749.25	743.32	737.96	0.68	1.36	0.24	0.22	0.01	86	330	148
Av. 200–500	1515.61	1490.12	1474.25	1.72	2.77	16.33	14.36	0.31	412	2047	762
Av. 600–1000	2286.76	2242.82	2212.61	1.98	3.34	191.25	154.63	1.95	1007	5669	1952

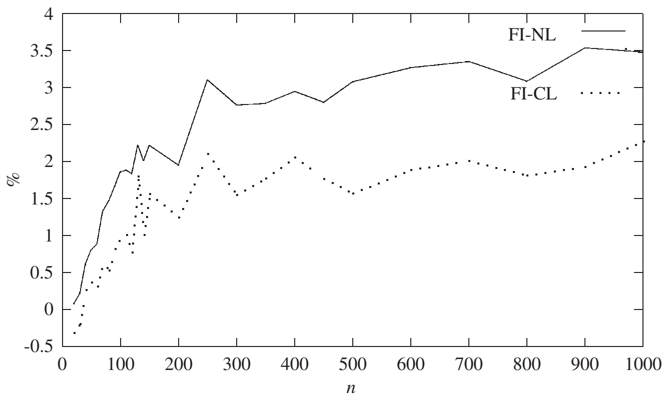

Fig. 5. Average difference in results of FI-NL and FI-CL over BI-NL for random Euclidean instances; % improvements are defined as $(v(\text{BI}) - v(\text{FI}))/v(\text{FI}) \times 100$.

Table 2
Average results on 10 random initial permutations for Euclidean TSP instances from TSP-LIB

Problem name	Optimal solution	% error			CPU times			# of iterations		
		BI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL
eil51	429.98	5.36	5.16	2.95	0.04	0.01	0.01	47	162	74
berlin52	7544.37	8.27	7.99	6.00	0.03	0.01	0.01	49	154	91
kroC100	20749.00	6.27	4.36	5.42	0.22	0.09	0.02	107	333	182
kroD100	21294.00	7.24	5.62	6.19	0.23	0.24	0.02	100	404	244
lin105	14379.00	5.66	5.66	5.34	0.24	0.27	0.02	114	445	208
ch130	6110.86	7.96	7.48	4.52	0.54	0.49	0.04	137	631	295
ch150	6532.28	10.52	7.87	5.23	0.82	0.73	0.05	155	666	305
d198	15780.00	4.19	4.36	3.93	2.16	2.85	0.11	221	954	464
tsp225	3859.00	9.39	8.95	5.99	2.91	2.77	0.11	245	1203	479
pr299	48191.00	9.80	7.89	6.52	7.35	9.39	0.20	344	1618	772
linhp318	41345.00	9.49	8.03	7.99	8.96	8.60	0.26	399	1779	810
rd400	15281.00	9.00	7.42	6.21	21.16	18.60	0.45	469	2482	988
pr439	107215.00	10.51	8.00	7.20	27.07	30.59	0.48	534	2798	895
pcb442	50779.00	10.68	8.80	5.76	26.75	20.16	0.50	513	2437	913
u574	36905.00	9.65	8.31	7.02	60.13	49.99	0.87	725	3830	1519
p654	34643.00	6.98	4.93	6.15	88.82	150.32	1.17	801	3939	1448
pr1002	259045.00	10.04	9.28	7.58	343.82	246.63	2.74	1299	7244	3029
u1060	224094.00	9.53	8.78	7.59	407.09	355.67	3.17	1401	8114	3048
pcb1173	56892.00	12.61	9.54	7.39	627.97	376.06	3.81	1523	8848	3175
d1291	50606.00	13.29	11.82	8.96	871.49	692.23	4.23	1756	9248	2766
rl1323	269554.00	12.27	8.99	6.72	952.32	853.55	4.50	1849	9988	2918
u1432	152970.00	11.99	9.76	6.86	1161.29	583.91	5.38	1713	10781	3464
pr2392	378032.00	12.65	10.10	9.00	4979.86	1951.96	16.02	3292	18342	5857

(viii) While ranking of heuristics by number of iterations required remains the same as in the Euclidean case, differences are less pronounced: FI-CL takes about 4 times the number of iterations of BI-NL (instead of about 6 times) and FI-NL roughly 1.5 times that number (instead of about 2.5 times).

4. Further experiments and explanations

Reason for the surprising phenomenon described in the previous section, i.e., that best improvement gives worse results than first improvement, is not immediately apparent. Observe that there might be several, as explanations for the good performance of FI-CL and FI-NL could be different. Indeed, their results are similar for the Euclidean case, but not for random distances. Several further series of experiments were conducted to find adequate explanations.

4.1. Improvement directions

A first series aimed at determining if it was beneficial or not to use a constant target value (in terms of the interval between best and worst improvement) for the improvement at each iteration. Therefore heuristic BI-NL was modified as follows: in a first pass, at each iteration, the best improvement $\Delta_{\max}$ and worst improvement $\Delta_{\min}$ are determined; a target value for improvement

$$\Delta_p = p\Delta_{\min} + (1-p)\Delta_{\max} \tag{1}$$

is chosen, where $p \in [0, 1]$ is a given parameter; in a second pass the improvement closest in absolute value to Δ_t is determined and current solution updated accordingly.

Results for $n = 100$ to $n = 300$, and p equal to 0.00, 0.05, 0.10, 0.15, ..., 1.00 are presented in Fig. 7. Values are averages for the same 100 Euclidean TSP problems of each size. It appears that:

(i) No precise value of p appears to be better than all others in all cases.

Table 3
Average results on 1000 random matrix distance instances for $n = 20, \dots, 150$, and 100 instances for $n = 200, \dots, 1000$

n	Obj. function values			% improv.		CPU times			# of iterations		
	BI-NL	FI-CL	FI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL	BI-NL	FI-CL	FI-NL
20	229.83	236.90	224.51	−3.15	2.37	0.00	0.00	0.00	14	34	23
30	255.36	263.12	242.85	−3.20	5.15	0.00	0.00	0.00	23	61	38
40	276.74	284.92	256.77	−3.19	7.78	0.01	0.00	0.00	33	91	54
50	293.80	301.83	266.11	−3.02	10.41	0.03	0.01	0.00	42	123	71
60	309.20	320.76	274.15	−4.22	12.78	0.06	0.01	0.01	52	156	87
70	324.51	335.13	282.45	−3.76	14.89	0.11	0.03	0.01	62	190	104
80	337.56	347.00	289.06	−3.27	16.78	0.16	0.04	0.01	72	226	121
90	347.36	358.65	293.12	−3.85	18.50	0.23	0.05	0.02	82	263	138
100	359.88	370.84	300.03	−3.65	19.95	0.33	0.07	0.02	93	300	155
110	373.09	382.35	305.90	−3.03	21.96	0.44	0.09	0.03	103	338	173
120	382.59	394.12	311.21	−3.70	22.94	0.57	0.11	0.03	114	377	191
130	389.94	403.87	315.47	−4.42	23.61	0.73	0.13	0.04	125	418	208
140	403.37	414.44	320.83	−3.45	25.73	0.91	0.15	0.05	135	459	226
150	411.61	423.14	325.60	−3.54	26.42	1.12	0.18	0.05	147	497	245
200	456.07	468.96	346.95	−3.72	31.45	2.37	0.31	0.09	202	706	337
250	493.86	505.77	365.28	−3.26	35.20	4.85	0.56	0.13	258	929	428
300	530.36	544.32	377.13	−3.70	40.63	8.98	0.85	0.20	313	1163	521
350	555.99	578.67	393.50	−5.76	41.29	14.46	1.29	0.28	374	1393	614
400	590.03	605.46	405.71	−3.80	45.43	22.29	1.81	0.38	433	1630	708
450	619.18	637.99	420.02	−4.48	47.42	32.71	2.33	0.48	491	1878	806
500	643.11	667.92	431.75	−5.75	48.95	46.11	3.14	0.60	552	2124	893
600	697.79	719.37	451.09	−4.78	54.69	91.95	5.55	0.98	670	2638	1091
700	743.42	764.78	468.49	−4.56	58.68	148.14	8.19	1.35	787	3155	1286
800	786.92	811.33	486.44	−5.02	61.77	222.78	11.51	1.77	914	3696	1481
900	827.16	856.07	504.28	−5.73	64.03	318.71	15.12	2.26	1039	4227	1685
1000	868.17	890.37	519.23	−4.28	67.20	440.23	19.86	2.80	1155	4792	1867
Av. 20–150	335.35	345.51	286.29	−3.53	16.38	0.34	0.06	0.02	78	252	131
Av. 200–500	555.51	572.73	391.48	−4.35	41.48	18.82	1.47	0.31	375	1403	615
Av. 600–1000	784.69	808.38	485.91	−4.87	61.28	244.36	12.05	1.83	913	3702	1482

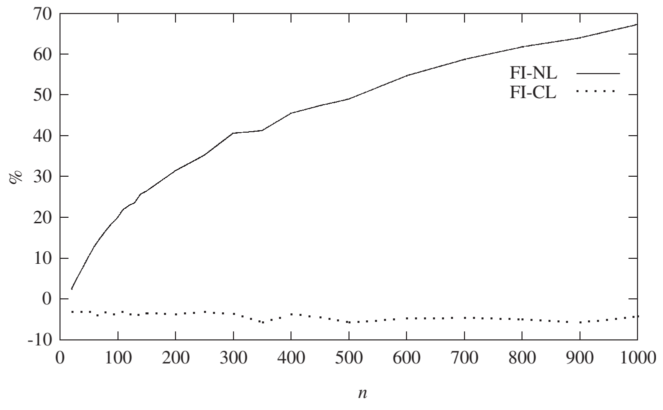

Fig. 6. Average % improvement (or deterioration) of FI-NL and FI-CL over BI-NL for random matrix instances.

(ii) Best values of p appear to be in the range $[0, 0.2]$, i.e. close to best improvement but not necessary at it.
(iii) Computing times and number of iterations augment moderately when p increases, and very substantially when $p = 1$ (therefore not all problems could be solved in reasonable time for that value).

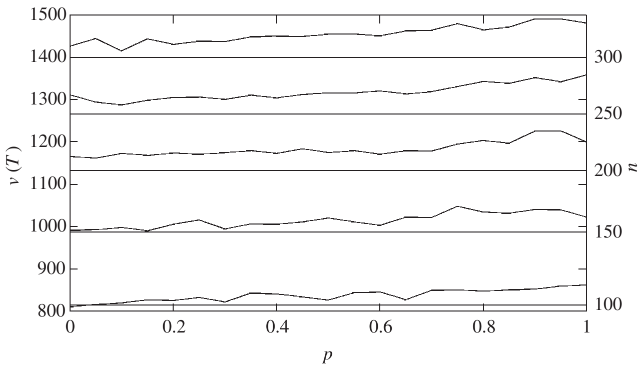

Fig. 7. Average objective values on 100 random *Euclidean* instances for $n = 100, 150, 200, 250$ and 300 as a function of target value p ($p = 0$ and $p = 1$ correspond to best and worst improvement 2-opt versions, respectively); horizontal lines represent average value obtained by FI-NL.

In conclusion, this series of experiments shows that slope of the descent does not play an important role in the performance of 2-opt for Euclidean TSP as long as it remains moderate. Similar conclusions were obtained for random distances.

4.2. *First improvement exchange*

Three further series of experiments study more closely how the heuristics FI-CL and FI-NL go from one iteration to the next. The versions described in the previous section continue from the current city in the tour after each iteration (a version noted CURR in the following tables). Two other options are to return to the initial city (noted BACK) or choose at random the position from where to test for improving exchanges (a version noted RAND). Results for FI-CL and the same Euclidean TSPs as in Table 1 are presented in Table 4. It appears that, for the Euclidean TSPs:

(i) Results obtained with the RAND version of FI-CL are worse than those of the two other versions; they are only slightly better than the results obtained with BI-NL.
(ii) Results obtained with the CURR version of FI-CL are worse than those of the BACK version, by about 0.4% for the larger problems.
(iii) The RAND version of FI-CL is much faster than the other two and performs less iterations; thus, it finds larger moves in less time (its average CPU time per iteration for instances with $n = 600, \ldots, 1000$ is 0.005 s, while CURR and BACK spend 0.027 and 0.088 s per iteration, respectively), but with the worst final solution quality; we conclude that systematic search for successive improving exchanges is more time consuming, but more effective than random search.

Results of similar experiments with the neighbor list implementation FI-NL are given in Table 5. It appears that:

(i) Ranking of the three versions is the same as for FI-CL.
(ii) Performances are always substantially better than for BI-NL; in other words, differences between the three versions are smaller than with FI-CL.

Results with the three versions of FI-CL applied to TSPs with random data are given in Table 6. It appears that:

(i) Results are less clear-cut than with Euclidean TSPs: no version is uniformly better than another one.
(ii) The ranking of versions appears to be again the same, but with smaller differences.

Another way to see if there is a significant difference between the three first improvement and the best improvement methods is to apply them a certain number of times on the same instance, but with different initial tours. We compare

Table 4
Average results for Euclidean TSP on 1000 random instances for $n = 20, \ldots, 150$, and 100 random instances for $n = 200, \ldots, 1000$, by variants of 2-opt and classical implementation

n	Obj. function values			% improv.		CPU times			# of iterations		
	RAND	CURR	BACK	CURR	BACK	RAND	CURR	BACK	RAND	CURR	BACK
20	397.25	395.48	396.72	0.45	0.13	0.00	0.00	0.00	38	37	37
30	479.31	474.80	474.90	0.95	0.93	0.00	0.00	0.00	69	69	70
40	548.37	540.52	541.79	1.45	1.22	0.00	0.01	0.01	106	107	106
50	609.59	601.35	601.01	1.37	1.43	0.01	0.02	0.02	146	149	149
60	662.74	652.68	651.61	1.54	1.71	0.01	0.03	0.03	188	192	192
70	711.44	700.05	699.30	1.63	1.74	0.02	0.05	0.06	231	239	239
80	757.54	747.04	745.03	1.41	1.68	0.02	0.07	0.09	278	288	288
90	802.71	790.03	788.14	1.61	1.85	0.04	0.12	0.16	325	338	338
100	843.34	828.75	829.21	1.76	1.70	0.05	0.19	0.26	375	391	389
110	880.47	864.90	864.49	1.80	1.85	0.07	0.27	0.38	424	445	445
120	920.15	903.99	902.03	1.79	2.01	0.08	0.37	0.53	473	502	500
130	948.43	932.20	935.28	1.74	1.41	0.11	0.46	0.69	528	561	552
140	994.62	974.02	972.03	2.12	2.32	0.14	0.62	0.97	581	619	617
150	1028.75	1000.62	1003.01	2.81	2.57	0.16	0.81	1.25	633	682	679
200	1176.75	1152.36	1149.67	2.12	2.36	0.37	1.95	3.48	910	988	979
250	1303.14	1279.39	1282.11	1.86	1.64	0.70	4.00	7.62	1215	1311	1307
300	1426.20	1400.41	1399.14	1.84	1.93	1.16	6.59	14.69	1518	1659	1652
350	1532.19	1504.22	1503.01	1.86	1.94	1.84	11.30	23.99	1835	2015	2020
400	1632.69	1603.03	1603.25	1.85	1.84	2.89	16.75	40.44	2154	2387	2396
450	1726.32	1697.76	1699.09	1.68	1.60	3.93	25.71	52.70	2490	2793	2787
500	1818.34	1793.68	1786.07	1.38	1.81	5.55	34.23	79.21	2837	3172	3177
600	1993.66	1958.54	1949.44	1.80	2.27	9.88	60.72	164.52	3531	3949	3979
700	2148.34	2111.12	2101.60	1.77	2.22	15.40	96.51	281.23	4247	4801	4837
800	2284.87	2252.68	2241.96	1.44	1.91	23.80	141.41	438.78	4999	5663	5734
900	2422.80	2383.16	2374.97	1.67	2.01	34.05	203.08	620.51	5750	6526	6668
1000	2550.02	2508.58	2502.33	1.66	1.91	46.03	271.42	1041.04	6497	7407	7559
Av. 20–150	756.05	743.32	743.18	1.60	1.61	0.05	0.22	0.32	314	330	329
Av. 200–500	1516.52	1490.12	1488.91	1.80	1.87	2.35	14.36	31.73	1851	2047	2045
Av. 600–1000	2279.94	2242.82	2234.06	1.67	2.07	25.83	154.63	509.22	5005	5669	5755

the three FI-NL versions by solving a random Euclidean instance with $n = 500$. The empirical distributions of the tour lengths of all four variants obtained after generating 1000 initial solutions are given in Fig. 8, from where similar conclusions as before can be derived: the best performance is that of the BACK version of first improvement and the worst one that of best improvement.

In conclusion, coordination between iterations appears to play a substantial role in the performance of FI-CL and FI-NL in the Euclidean case; it appears to be much less important in the case of random distances.

4.3. Ranks of edges

In the last series of experiments we investigate if the ranks of entering and leaving edges as well as average rank of edges in the current solution give possible explanations of why first improvement 2-opt exchange is better than best improvement.

Assume that matrices D', R and P (see Step 1 in Fig. 3) are known. We first define the *rank of an edge* (i, j) as

$$r(i, j) = \frac{p_{i,j} + p_{j,i}}{2}, \qquad (2)$$

i.e., the rank is defined as the average position of cities with index j (in the ranked list of cities with index i) and i (in the ranked list of cities with index j). Then the random variable *rank of the tour* is defined as the average sum of ranks

Table 5
Average results for Euclidean TSP on 1000 random instances for $n = 20, \ldots, 150$, and 100 random instances for $n = 200, \ldots, 1000$, by variants of 2-opt and *neighbor list* implementation

n	Obj. function values			% improv.		CPU times			# of iterations		
	RAND	CURR	BACK	CURR	BACK	RAND	CURR	BACK	RAND	CURR	BACK
20	394.22	393.94	394.02	0.07	0.05	0.00	0.00	0.00	23	24	23
30	473.78	472.85	473.58	0.20	0.04	0.00	0.00	0.00	39	41	38
40	539.58	538.59	537.48	0.18	0.39	0.00	0.00	0.00	56	58	55
50	599.20	598.66	598.39	0.09	0.14	0.00	0.00	0.00	74	77	73
60	649.49	648.95	647.62	0.08	0.29	0.01	0.00	0.01	93	95	91
70	695.65	695.15	693.62	0.07	0.29	0.01	0.01	0.01	112	115	110
80	740.65	740.06	739.76	0.08	0.12	0.01	0.01	0.02	131	135	128
90	783.56	783.40	781.17	0.02	0.31	0.02	0.01	0.02	151	154	149
100	823.55	821.42	820.76	0.26	0.34	0.02	0.01	0.03	171	176	169
110	857.85	857.34	855.76	0.06	0.24	0.03	0.02	0.03	191	197	190
120	893.80	894.46	893.42	−0.07	0.04	0.03	0.02	0.04	212	217	211
130	930.15	928.20	926.16	0.21	0.43	0.04	0.02	0.05	234	241	233
140	965.69	964.32	960.80	0.14	0.51	0.05	0.03	0.06	257	261	253
150	995.12	994.05	992.46	0.11	0.27	0.06	0.04	0.06	276	283	275
200	1143.28	1144.30	1139.38	−0.09	0.34	0.10	0.07	0.12	383	397	388
250	1273.12	1266.69	1265.91	0.51	0.57	0.17	0.12	0.19	504	520	498
300	1382.48	1383.51	1378.89	−0.07	0.26	0.25	0.19	0.29	622	630	632
350	1486.15	1489.10	1480.06	−0.20	0.41	0.35	0.27	0.40	751	750	733
400	1586.56	1588.80	1581.82	−0.14	0.30	0.48	0.38	0.55	880	892	881
450	1679.70	1680.46	1676.93	−0.05	0.17	0.62	0.49	0.72	1006	1008	987
500	1771.05	1766.90	1766.58	0.23	0.25	0.78	0.63	0.89	1126	1138	1108
600	1934.24	1931.68	1934.25	0.13	0.00	1.17	1.00	1.35	1411	1410	1384
700	2084.33	2083.07	2080.31	0.06	0.19	1.64	1.41	1.90	1666	1660	1671
800	2223.34	2224.27	2221.12	−0.04	0.10	2.18	1.87	2.56	1949	1950	1937
900	2354.88	2345.38	2348.42	0.41	0.28	2.84	2.42	3.24	2248	2236	2206
1000	2479.90	2478.63	2475.53	0.05	0.18	3.58	3.03	4.13	2529	2505	2517
Av. 20–150	738.73	737.96	736.79	0.11	0.25	0.02	0.01	0.02	144	148	143
Av. 200–500	1474.62	1474.25	1469.94	0.03	0.33	0.39	0.31	0.45	753	762	747
Av. 600–1000	2215.34	2212.61	2211.93	0.12	0.15	2.28	1.95	2.64	1961	1952	1943

of the edges in the tour $T = (x_i)$, $i = 1, \ldots, n$,

$$r(T) = \frac{1}{n} \sum_{i=1}^{n} r(x_i, x_{i+1}), \tag{3}$$

where $x_{n+1} = x_1$.

As in Step 3 of Figs. 1–4, let us denote by $e_1 = (x_i, x_j)$ and $e_2 = (x_{i+1}, x_{j+1})$ the two edges that enter the solution in some iteration of 2-opt and by $e_3 = (x_i, x_{i+1})$ and $e_4 = (x_j, x_{j+1})$ the two leaving edges. Let us further define random variables that represent smaller and larger ranks of entering or leaving edges:

$$\alpha_1 = \min\{r(e_1), r(e_2)\}; \quad \alpha_2 = \max\{r(e_1), r(e_2)\};$$

$$\alpha_3 = \min\{r(e_3), r(e_4)\}; \quad \alpha_4 = \max\{r(e_3), r(e_4)\};$$

In our tests we found experimentally approximate expected values of α_j, $j = 1, 2, 3, 4$ for each variant of 2-opt mentioned before. In Table 7 we present some of the results for 10 random Euclidean distance problems with $n = 500$. Let us denote by $\alpha_j(\text{BI-NL})$, $\alpha_j(\text{FI-NL})$ and $\alpha_j(\text{FI-NL})$ values obtained by BI-NL, FI-NL and FI-CL, respectively.

Table 6
Average results on 1000 random matrix instances for $n = 20, \ldots, 150$, and 100 distances for $n = 200, \ldots, 1000$, by variants of 2-opt and classical implementation

n	Obj. function values			% improv.		CPU times			# of iterations		
	RAND	CURR	BACK	CURR	BACK	RAND	CURR	BACK	RAND	CURR	BACK
20	237.44	236.90	236.72	0.23	0.30	0.00	0.00	0.00	32	34	36
30	262.24	263.12	265.29	−0.34	−1.16	0.00	0.00	0.00	57	61	63
40	287.53	284.92	287.60	0.92	−0.02	0.00	0.00	0.00	85	91	94
50	303.39	301.83	304.63	0.52	−0.41	0.01	0.01	0.01	114	123	126
60	319.47	320.76	319.77	−0.40	−0.09	0.01	0.01	0.02	145	156	161
70	336.23	335.13	334.65	0.33	0.47	0.02	0.03	0.03	176	190	196
80	349.32	347.00	348.01	0.67	0.38	0.03	0.04	0.05	208	226	232
90	358.81	358.65	359.46	0.04	−0.18	0.04	0.05	0.07	244	263	271
100	372.17	370.84	369.80	0.36	0.64	0.05	0.07	0.10	277	300	309
110	384.99	382.35	383.38	0.69	0.42	0.06	0.09	0.12	310	338	349
120	393.87	394.12	395.64	−0.06	−0.45	0.07	0.11	0.15	345	377	388
130	404.37	403.87	402.57	0.12	0.45	0.09	0.13	0.19	380	418	430
140	416.91	414.44	414.71	0.60	0.53	0.11	0.16	0.24	416	459	471
150	426.87	423.14	424.49	0.88	0.56	0.13	0.18	0.28	453	497	513
200	468.01	468.96	470.34	−0.20	−0.50	0.24	0.36	0.61	639	706	726
250	512.18	505.77	506.01	1.27	1.22	0.42	0.64	1.11	832	929	956
300	546.28	544.32	543.71	0.36	0.47	0.68	0.96	1.83	1039	1163	1193
350	581.90	578.67	576.67	0.56	0.91	0.99	1.49	2.86	1236	1393	1433
400	609.62	605.46	614.08	0.69	−0.74	1.36	2.07	4.08	1443	1630	1674
450	641.07	637.99	641.54	0.48	−0.07	1.82	2.69	5.58	1653	1878	1930
500	665.63	667.92	669.73	−0.34	−0.62	2.40	3.64	7.70	1872	2124	2176
600	725.27	719.37	716.52	0.82	1.22	3.67	5.56	12.67	2306	2638	2722
700	769.07	764.78	765.71	0.56	0.44	5.51	8.22	19.65	2762	3155	3242
800	816.14	811.33	810.79	0.59	0.66	7.40	11.54	28.27	3203	3696	3798
900	858.83	856.07	853.67	0.32	0.60	9.64	15.08	38.69	3664	4227	4357
1000	898.16	890.37	894.32	0.87	0.43	12.64	19.78	50.80	4132	4792	4931
Av. 20–150	346.69	345.51	346.19	0.33	0.10	0.04	0.06	0.09	232	252	260
Av. 200–500	574.96	572.73	574.58	0.40	0.10	1.13	1.69	3.40	1245	1403	1441
Av. 600–1000	813.49	808.38	808.20	0.64	0.67	7.77	12.04	30.02	3213	3702	3810

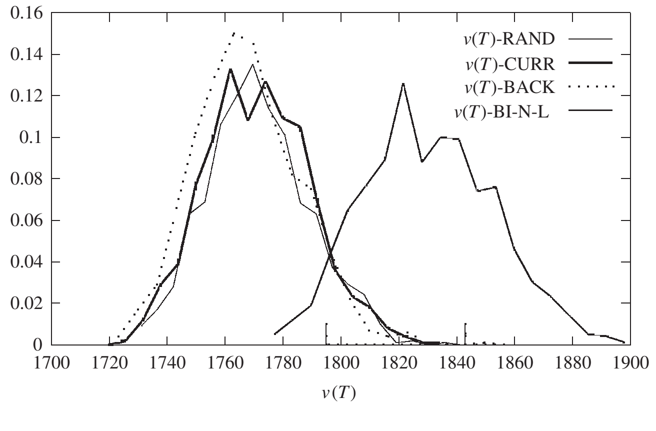

Fig. 8. Empirical distributions of r.v. $v(T)$ obtained by 1000 restarts of the three FI-NL versions and BI-NL in solving one random Euclidean instance ($n = 500$).

Table 7
Average ranks of entering edges in 10 random Euclidean instances obtained by different 2-opt versions

Pr.	BI-NL			FI-NL (CURR)			FI-CL (RAND)			FI-CL (CURR)			FI-CL (BACK)		
#	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$
1	5.1	15.9	1842.9	3.7	181.9	1797.2	31.1	99.1	1819.4	29.8	119.8	1765.6	27.9	115.5	1797.2
2	5.1	14.8	1821.6	3.9	184.8	1787.6	31.0	98.7	1821.0	30.5	104.7	1794.0	30.6	120.0	1775.5
3	5.0	15.3	1809.3	3.9	164.7	1775.4	30.7	100.9	1824.3	30.3	120.9	1786.5	26.8	117.4	1759.2
4	5.1	15.4	1816.5	3.8	194.3	1786.8	29.2	94.1	1804.5	28.6	112.4	1806.8	29.3	119.9	1792.6
5	5.0	15.4	1841.1	3.9	171.7	1808.1	29.6	98.9	1818.6	28.9	119.7	1818.1	29.5	121.5	1770.0
6	5.2	15.3	1860.3	3.6	197.5	1790.9	31.5	98.0	1804.3	28.8	115.7	1826.4	27.8	119.6	1818.9
7	5.1	15.6	1743.7	3.9	160.5	1706.9	29.0	94.6	1744.4	28.3	110.4	1738.8	28.7	114.5	1754.0
8	5.0	15.9	1860.7	3.8	184.3	1762.7	29.6	99.3	1841.7	29.4	112.3	1784.5	29.4	120.3	1776.8
9	5.3	14.7	1853.3	3.9	182.5	1748.8	30.0	100.3	1851.0	30.1	114.0	1784.2	31.1	121.9	1761.1
10	5.1	16.1	1812.9	3.9	168.8	1803.3	29.0	93.1	1789.2	28.3	110.1	1764.1	28.1	113.7	1797.0
Av.	5.1	15.5	1826.2	3.8	179.0	1776.5	30.1	97.7	1811.9	29.3	114.0	1786.9	28.9	118.6	1780.2

From Table 7, the following conclusions can be drawn:

(i) Average rank of smaller entering edges (α_1) is always smaller when applying FI-NL than with BI-NL. Moreover, the following holds:

$$\alpha_1(\text{FI-NL}) \leqslant \alpha_1(\text{BI-NL}) \leqslant \alpha_2(\text{BI-NL}) \leqslant \alpha_2(\text{FI-NL}), \tag{4}$$

i.e., average ranks of entering edges for FI-NL bracket those for BI-NL. From this inequality we conclude that *in best improvement both entering edges are small and such small edges tend to remain in the tour at further iterations*, while *FI-NL leaves in the solution very small edges as well as large edges*, *which can be easily removed later*.

(ii) The average quality of the solutions obtained by the three FI-CL variants are ranked in the same way as the corresponding values of α_1, i.e.,

$$\alpha_1(\text{FI-CL-B}) \leqslant \alpha_1(\text{FI-CL-C}) \leqslant \alpha_1(\text{FI-CL-R}). \tag{5}$$

These values are much larger than for the neighbor list implementation.

(iii) The larger the average value of α_2, the better is the final tour obtained. The ranking is

$$\alpha_2(\text{FI-NL}) \geqslant \alpha_2(\text{FI-CL-B}) \geqslant \alpha_2(\text{FI-CL-C}) \geqslant \alpha_2(\text{FI-CL-R}) \geqslant \alpha_2(\text{BI-NL}). \tag{6}$$

This tends to confirm the advantage of being able to remove easily the large edges introduced in the tour.

In Table 8 the first problem from Table 7 is analyzed in more detail. Each line represents average results in the previous 100, 200 and 500 iterations obtained by BI-NL, FI-NL and FI-CL, respectively. The second column gives number of iterations done. Columns 3 and 4 report average values of indices of cities i and j when improvement is made. Columns 5–8 give values of α_j, $j = 1, 2, 3, 4$, obtained in last 100, 200 or 500 iterations (depending on the method used), while columns 9–12 report average results from the first iteration during the current phase. In columns 13 and 14, average rank of last 100, 200 or 500 solutions and standard deviation from it are reported, for each phase. The last column gives average improvement per iteration. It should be noted that general conclusions cannot be derived from Table 8 since only one problem is considered. For example, the solution obtained by FI-CL (CURR) is better than that of FI-NL, which is not the case on average (see Table 1). Anyway, some trends are typical and easy to recognize:

(i) The outer loop index i (or position in the tour of city i) when a move takes place has a value very close to the average one (e.g., $n/2 = 250$) only for FI-CL (RAND). That was to be expected since i in the RAND version is chosen at random; this index monotonously increases for BACK version of FI-CL, i.e., with this version smallest edges are introduced first, together with large ones, then slightly larger ones.

Table 8
Statistics on one $n = 500$ random Euclidean instance solved by different 2-opt versions

Mth.	Iter.	i	j	α_1	α_2	α_3	α_4	α_1	α_2	α_3	α_4	$r(T)$	St.d	$v(T)$	Dif-a
	100	159.8	332.2	8.2	28.0	373.5	404.4	8.2	28.0	373.5	404.4	101.7	102.7	12323.2	−133.3
BI	200	140.1	302.6	8.3	28.6	186.7	225.0	8.2	28.3	280.1	314.7	26.7	31.2	5798.9	−65.2
	300	177.0	310.1	4.9	17.6	37.7	67.2	7.1	24.7	199.3	232.2	10.3	9.2	3453.1	−23.5
NL	400	210.3	298.4	3.5	7.8	11.3	21.7	6.2	20.5	152.3	179.6	5.9	4.4	2454.9	−10.0
	500	242.1	265.0	3.1	6.3	6.0	11.5	5.6	17.6	123.1	146.0	4.3	2.9	1995.0	−4.6
	596	239.6	270.2	3.2	6.7	4.2	8.2	5.2	15.9	103.9	123.8	3.8	2.3	1842.9	−1.6
	200	68.4	236.4	3.6	212.0	117.4	268.3	3.6	212.0	117.4	268.3	182.0	164.2	19404.5	−31.2
FI	400	84.6	224.1	3.7	205.0	99.0	261.0	3.7	208.5	108.2	264.6	121.5	156.3	13823.0	−27.9
	600	192.3	305.3	3.7	217.8	106.5	273.3	3.7	211.6	107.6	267.5	58.2	120.8	7403.3	−32.1
NL	800	161.1	255.6	4.1	169.1	60.9	215.4	3.8	201.0	95.9	254.5	17.0	57.0	3153.1	−21.3
	969	336.1	369.8	4.5	91.6	18.5	117.2	3.9	181.9	82.4	230.5	3.6	2.3	1797.2	−8.0
	500	257.3	9.1	87.8	249.2	140.9	315.5	87.8	249.2	140.9	315.5	130.7	111.6	15511.9	−20.3
FI	1000	247.4	20.5	45.7	152.4	69.8	193.4	66.7	200.8	105.4	254.4	65.6	68.4	9844.2	−11.3
	1500	245.5	41.3	22.4	79.9	33.2	103.8	52.0	160.5	81.3	204.2	31.0	34.6	6216.8	−7.3
CL	2000	250.4	97.9	10.0	43.1	14.3	54.2	41.5	131.2	64.5	166.7	15.6	19.9	4113.7	−4.2
	2500	239.7	144.5	6.4	23.4	8.9	29.9	34.4	109.6	53.4	139.3	6.6	7.6	2472.1	−3.3
RAND	2798	250.1	186.1	4.3	11.5	5.8	14.8	31.2	99.1	48.3	126.1	3.7	2.3	1819.4	−2.2
	500	4.7	94.0	57.7	220.3	94.1	259.3	57.7	220.3	94.1	259.3	174.7	132.9	18941.8	−13.4
FI	1000	12.6	111.5	42.6	180.2	73.9	209.6	50.1	200.2	84.0	234.4	113.9	112.9	13619.9	−10.6
	1500	37.6	115.3	29.5	118.5	50.3	136.3	43.3	173.0	72.8	201.7	75.4	91.3	10016.7	−7.2
CL	2000	36.2	142.6	29.5	119.3	46.4	135.9	39.8	159.6	66.2	185.3	41.8	59.7	6825.0	−6.4
	2500	158.4	220.5	15.1	62.0	22.6	72.8	34.9	140.0	57.5	162.8	23.5	42.7	4668.5	−4.3
CURR	3000	53.0	227.3	13.7	54.3	20.7	63.0	31.3	125.8	51.3	146.1	7.7	17.1	2463.0	−4.4
	3211	170.6	315.2	7.1	34.4	11.2	40.1	29.8	119.8	48.7	139.2	3.5	1.9	1765.6	−3.3
	500	2.5	52.6	63.1	239.7	109.0	284.8	63.1	239.7	109.0	284.8	159.1	129.5	17606.2	−16.1
FI	1000	22.1	82.1	44.1	173.3	78.2	198.2	53.6	206.5	93.6	241.5	100.1	106.9	12363.5	−10.5
	1500	55.4	118.0	26.5	108.6	41.3	127.9	44.6	173.9	76.2	203.6	66.0	82.1	9209.2	−6.3
CL	2000	86.7	137.2	22.7	99.4	35.7	113.1	39.1	155.3	66.1	181.0	39.3	58.8	6562.9	−5.3
	2500	177.4	240.7	14.9	62.7	22.1	75.3	34.3	136.8	57.3	159.9	19.5	34.2	4244.7	−4.6
BACK	3000	284.9	315.3	9.7	45.5	15.5	53.1	30.2	121.6	50.3	142.1	6.2	9.8	2304.1	−3.9
	3185	342.0	385.7	5.5	17.0	7.4	21.7	28.7	115.5	47.8	135.1	3.7	2.5	1797.2	−2.7

(ii) The rank of entering edges (e.g. value of α_1) decreases along iterations for all methods except for the best one (in average) FI-NL; by applying FI-NL exchanges, a solution is built by entering one very small and one large edge.
(iii) If we look at values of statistics α_2, α_3 and α_4 (averages in last 100, 200 or 500 iterations in columns 6–8, or averages in all iterations, columns 10–12), again they are strictly decreasing for all methods except FI-NL.
(iv) There is a strong correlation between values $r(T)$ and $v(T)$: the smaller the tour length, the smaller the rank of the tour.

The same experiments has been done for 10 random matrix instances. Results are presented in Table 9. We see that: (Table 10)

(i) Relation (4) still holds, but the difference between the first two statistics are much larger (compare $5.1 - 3.8 = 1.3$ from Table 7 with $9.4 - 5.2 = 4.2$ from Table 9).
(ii) Relation (5) holds as well, but CURR version of FI-CL is not the worst as for Euclidean problems.
(iii) The differences in values of α_2 for the three FI-CL 2-opt versions are small and no conclusion (as in (6)) can be made from this small sample.

Table 9
Ranks of entering edges in 10 random matrix instances obtained by different 2-opt versions

Pr.	BI-NL			FI-NL (CURR)			FI-CL (RAND)			FI-CL (CURR)			FI-CL (BACK)		
#	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$	α_1	α_2	$v(T)$
1	9.6	25.7	641.7	5.3	118.7	456.7	37.2	105.4	646.3	34.9	100.7	664.0	32.8	94.8	664.2
2	9.4	25.5	620.9	5.0	136.0	430.7	35.9	98.6	704.4	32.3	96.9	663.8	30.7	94.5	700.6
3	9.6	24.5	627.7	5.0	119.8	428.5	34.5	96.4	666.5	33.8	101.8	684.6	31.5	95.8	683.7
4	9.3	24.1	607.8	4.9	129.7	403.6	34.7	97.4	681.1	31.5	96.5	680.9	32.9	97.1	666.1
5	9.4	25.9	654.3	5.2	130.4	450.5	36.0	99.8	667.5	33.4	98.2	748.6	31.6	94.5	703.9
6	9.7	24.8	675.4	5.1	123.3	437.6	37.2	99.1	679.8	32.9	100.2	706.0	30.1	90.5	676.5
7	9.2	24.4	646.2	5.4	126.5	439.8	34.2	97.2	676.4	32.3	93.6	668.4	32.0	95.1	662.2
8	9.3	24.7	634.7	5.3	127.7	420.0	35.8	98.8	663.0	32.4	94.9	641.8	30.6	95.0	639.2
9	9.1	24.6	711.0	5.3	126.8	472.5	35.4	95.2	662.5	31.6	93.7	698.6	31.6	92.8	717.7
10	9.7	24.4	665.6	5.2	116.3	444.8	34.9	97.5	688.0	31.1	89.8	632.4	29.3	90.5	662.3
Av.	9.4	24.9	648.5	5.2	125.5	438.5	35.6	98.5	673.6	32.6	96.6	678.9	31.3	94.1	677.6

Table 10
Statistics on one $n = 500$ random matrix instance by different 2-opt versions

Mth.	Iter.	i	j	α_1	α_2	α_3	α_4	α_1	α_2	α_3	α_4	$r(T)$	St.d	$v(T)$	Dif-a
	100	158.1	325.5	13.8	39.0	391.1	425.4	13.8	39.0	391.1	425.4	110.2	101.5	10832.5	−153.2
BI	200	164.8	346.0	14.7	42.3	198.7	238.6	14.3	40.6	294.9	332.0	34.1	30.1	3227.1	−76.1
	300	164.6	344.5	9.3	21.5	46.0	75.4	12.6	34.3	212.0	246.5	16.0	11.5	1433.6	−17.9
NL	400	189.4	326.7	5.9	15.8	16.5	32.7	10.9	29.7	163.1	193.1	10.5	7.2	895.3	−5.4
	500	206.9	302.6	5.9	14.6	10.2	20.9	9.9	26.6	132.5	158.6	8.4	5.7	679.9	−2.2
	534	245.3	309.2	5.4	11.9	7.3	16.5	9.6	25.7	124.5	149.6	7.9	5.3	641.7	−1.1
	200	6.1	243.0	3.4	180.0	105.3	340.1	3.4	180.0	105.3	340.1	158.1	150.5	15664.1	−52.5
FI	400	11.5	232.7	4.2	156.9	70.6	275.1	3.8	168.4	88.0	307.6	84.2	121.7	8271.6	−37.0
	600	17.4	255.6	5.2	112.3	30.6	212.4	4.2	149.7	68.8	275.9	34.0	69.0	3247.2	−25.1
NL	800	35.4	276.2	6.9	58.9	19.5	110.8	4.9	127.0	56.5	234.6	8.2	15.0	684.5	−12.8
CURR	876	141.9	256.0	9.5	31.1	8.2	41.2	5.3	118.7	52.3	217.8	6.9	5.5	456.7	−2.5
	500	257.3	8.1	83.4	229.7	140.1	320.9	83.4	229.7	140.1	320.9	114.9	95.2	11321.5	−29.7
FI	1000	247.4	38.4	36.0	109.2	55.3	159.2	59.7	169.4	97.7	240.0	45.6	40.2	4361.1	−13.9
CL	1500	245.7	139.2	16.8	48.8	23.8	69.1	45.4	129.2	73.1	183.1	18.3	14.4	1663.3	−5.4
RAND	1923	249.0	232.5	7.9	20.8	12.6	28.4	37.2	105.4	59.8	149.0	7.9	5.0	646.3	−2.4
	500	1.0	25.5	60.9	183.3	82.7	258.8	60.9	183.3	82.7	258.8	165.6	143.8	16391.0	−19.5
FI	1000	1.0	54.0	51.8	146.7	64.4	221.5	56.3	165.0	73.5	240.2	78.2	75.2	7625.2	−17.5
	1500	1.8	82.0	28.1	81.0	37.8	116.9	46.9	137.0	61.6	199.1	32.6	27.1	3089.7	−9.1
CL	2000	4.8	195.9	12.0	31.5	16.7	46.1	38.2	110.6	50.4	160.8	13.2	9.9	1149.1	−3.9
CURR	2236	47.2	258.4	7.2	16.6	11.0	23.6	34.9	100.7	46.2	146.4	8.1	5.0	664.0	−2.1
	500	1.0	41.1	56.0	156.5	72.3	231.6	56.0	156.5	72.3	231.6	171.5	149.7	16946.2	−18.4
FI	1000	1.0	82.8	47.7	140.0	60.8	208.2	51.8	148.2	66.5	219.9	90.2	102.3	8866.3	−16.2
	1500	2.4	106.8	29.9	93.2	39.1	139.6	44.5	129.9	57.4	193.1	34.5	29.5	3275.7	−11.2
CL	2000	7.6	163.1	12.6	34.1	17.3	49.7	36.5	105.9	47.4	157.3	14.1	11.1	1245.7	−4.1
BACK	2291	49.1	237.6	7.3	18.3	11.2	24.7	32.8	94.8	42.8	140.4	8.2	5.3	664.2	−2.0

In some further experiments, a good initial solution was used, i.e. that one provided by a *nearest neighbor* or *greedy* heuristic. Then the improvements observed did not take place anymore. Moreover, first improvement was not only worse but also not faster than best improvement.

5. Conclusions

Numerous numerical studies of heuristics for the TSP have been made; however, they often concentrate on overall performance, instead of seeking insight into the heuristics behavior, i.e., finding precisely why some versions work better than others. In this paper it is shown how the well-known 2-opt heuristic exhibits unexpected behavior: the greedy or best improvement version yields substantially worse results than the first improvement version. This is true for the *classical* implementation which considers all possible exchanges at each iteration, or checks such exchanges until an improving one is found, as well as for the *neighbor list* implementation, which ranks edges and considers those incident to each node on the tour in turn.

Two factors appear to play a role in this phenomenon: (i) the coordination of iterations, where it is best, for Euclidean TSPs, to perform iterations from the beginning node and only proceed further when no improving moves can be made in that vicinity; (ii) the selection of both very short edges, which are likely to belong to optimal or near-optimal tours, together with longer edges which are likely to be eliminated at some further iteration, instead of pairs of short edges.

This suggests, when designing heuristics for combinatorial optimization problems, to look at conditions which make it likely for an element to belong to optimal or near-optimal solutions, and simple or composite moves which introduce, possibly with others, such elements in the current solution instead of second-best ones.

References

[1] J.L. Bentley, Fast algorithms for geometric traveling salesman problem, ORSA J. Comput. 4 (1992) 387–411.
[2] G.A. Croes, A method for solving traveling salesman problems, Oper. Res. 6 (1958) 791–812.
[3] M.M. Flood, The traveling salesman problem, Oper. Res. 4 (1) (1956) 61–75.
[4] M.L. Fredman, D.S. Johnson, L.A. McGeoch, G. Ostheimer, Data structures for traveling salesman, J. Algorithms 18 (1995) 432–479.
[5] P. Hansen, N. Mladenović, Variable neighborhood search: principles and applications, European J. Oper. Res. 130 (2001) 449–467.
[6] P. Hansen, N. Mladenovic, Variable neighbourhood search in: F. Glover, G. Kochenberger (Eds.), Handbook of Mataheuristics, Kluwer Academic Publishers, Dordrecht, 2003, pp. 145–184.
[7] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: a case study in local optimization, in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, New York, 1996.
[8] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc. 7 (1956) 48–50.
[9] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (1997) 1097–1100.
[10] G. Reinelt, TSP-LIB a traveling salesman library, ORSA J. Comput. 3 (1991) 376–384.
[11] K. Steiglitz, P. Weiner, Some improved algorithms for computer solution of the traveling salesman problem, in: Proceedings of the 6th Annual Alerton Conference on Communication, Control and Computing, University of Illinois, 1968, pp. 814–821.